CS 624: Analysis of Algorithms

Assignment 11

Due: Monday, May 3, 2021

As always, I'm looking for clear and careful explanations. Make it understandable. Make me happy I'm reading your paper!

1. Exercise 3.7 (page 16) in the lecture notes.

2. Here's a problem:

   Problem name: HITTING SET

   Instance: A collection C of subsets of a set S together with a positive integer K. (NOTE: The sets in the collection C may overlap. That's what gives this problem its complexity. If they don't overlap at all, the problem is trivial.)

   Question: Does S contain a *hitting set* for C of size K or less—that is, a subset $S' \subseteq S$ with $|S'| \leq K$ and such that S' contains at least one element of each set $c \in C$?

   Prove that HITTING SET is NP-complete.

   To do this you need to do two things:

   (a) Prove that HITTING SET is in NP. This should be extremely easy.

   (b) Prove that some problem that is already known to be NP-complete polynomially reduces to HITTING SET.

   Hint: In this case, I suggest you prove that VC $\leq_P$ HITTING SET.

3. It's natural to think that VERTEX COVER (or actually, the problem of finding a minimal vertex cover) can be dealt with by a greedy algorithm (which would almost certainly imply that VC would be in P). If this were true, then we would have proved that P = NP, which would be quite a remarkable result. The point of this problem is to show that this is quite unlikely.

   Here is a reasonable greedy algorithm for solving VC on a graph $G = (V, E)$. At each step in the algorithm, we will pick a vertex. And we will pick it to get as quickly as we can to a vertex cover—that's what makes this a "greedy" algorithm. We will add that vertex to the vertex cover (which is denoted by C, and is initialized to $\varnothing$), and then we will delete that vertex and all the edges incident on that vertex from G:

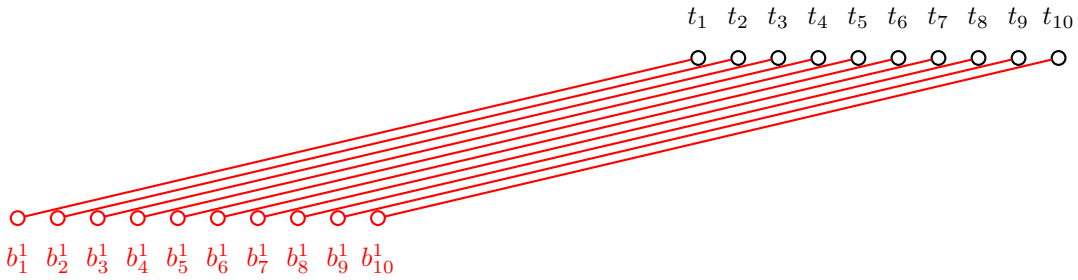
```
GREEDY-VC(G)
    C ← ∅
    while E ≠ ∅ do
        Pick a vertex v ∈ V having maximal degree in the current graph
        C ← C ∪ {v}
        V ← V \ {v}
        E ← E \ {e ∈ E : e is incident on v}
    return C
```

It's pretty clear that this does produce a vertex cover. Unfortunately though, it doesn't necessarily produce a minimal vertex cover. I'm going to give a famous example of how it can go wrong.

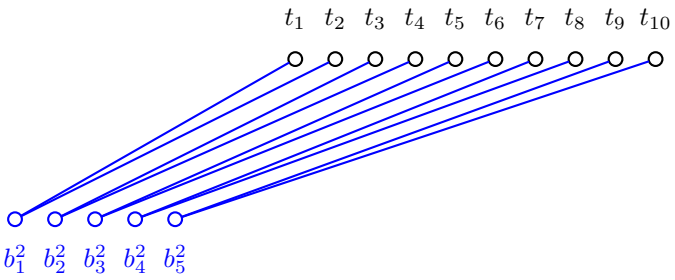
We start by creating a bipartite graph. The two "parts" of this graph will be the "top" vertices and the "bottom" vertices.

There are 10 top vertices. (We have chosen 10 just to make the example informative but not too complicated, but any number ≥ 2 could be chosen.)

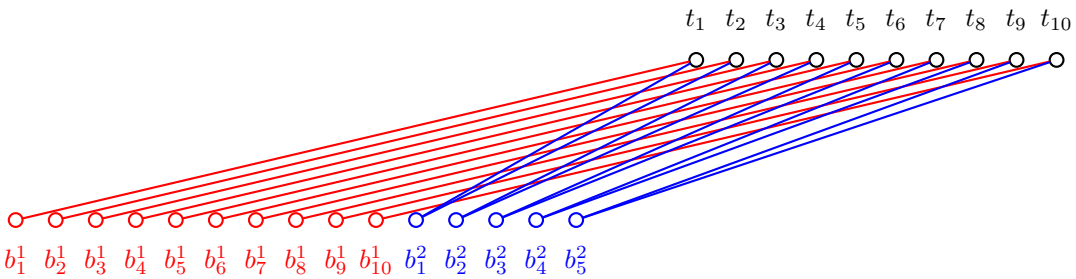
We start by creating a group (the "first" group) of 10 bottom vertices, and we create 1 edge from each bottom vertex to the corresponding top vertex, like this:



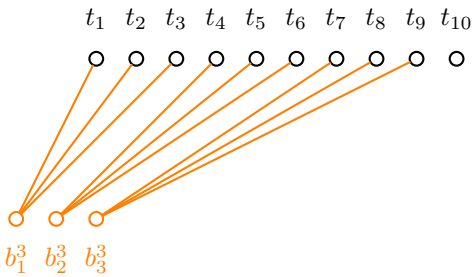
Next, we add a second group of bottom vertices. We connect each of these to *two* top vertices, in order. So the first new bottom vertex is connected to t_1 and t_2. The next new bottom vertex is connected to t_3 and t_4. And so on. We can create 5 bottom vertices in this way. The new bottom vertices thus look like this:



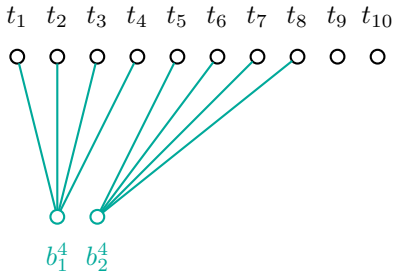
and so all together, we now have the following graph:



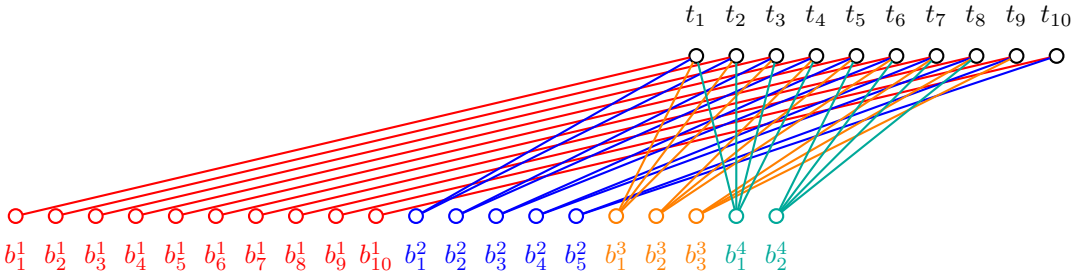
Next, we create a third group of bottom vertices. Each of these vertices is connected to *three* top vertices. So we will have three vertices in the third bottom group, and they will connect to all but the last top vertex:



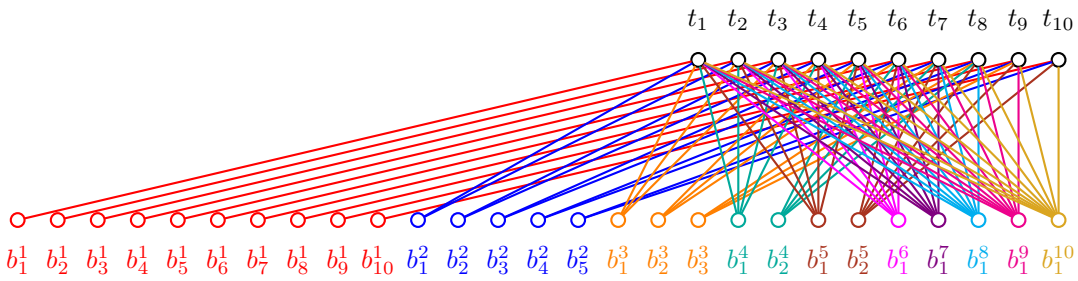
The fourth group of bottom vertices looks like this:



And so putting all these vertices together, we have this following graph:



We will continue like this, until we have 10 groups of bottom vertices. (You should be able to see easily why group 5 consists of two vertices, but groups 6–10 each consist of one vertex only.) The final graph we wind up with looks like this. It's not easy to see here what is really happening, but I think that based on the pictures above, you should be able to understand it.



(a) Prove that no vertex in G has more than 10 edges.

(b) Prove that if $1 < k \leq 10$ and if we remove all the bottom vertices in groups k up through group 10, then no vertex in G has more than $k - 1$ edges.

(c) Prove that the algorithm could therefore at each step pick the rightmost bottom vertex that has not yet been deleted, and add it to the vertex cover.

(d) This would yield a vertex cover consisting of all the bottom vertices. And that is indeed a vertex cover. However, since this is a bipartite graph, it is evident that the 10 top vertices constitute a smaller vertex cover. So the algorithm is not guaranteed to produce the smallest vertex cover.

The problem in this case is that there is a choice to be made of which vertex to pick next. In the case of this graph, picking the bottom vertex was acceptable according to the algorithm, but was not the best choice to make.

Using the same algorithm, what would be the best choice of vertices (acceptable to the algorithm) to make at each step for this graph?